This is Lab Worksheet 4 - not an Assignment

This Lab Worksheet contains some practical examples that will prepare you to complete your Assignments. You do **not** have to hand in this Lab Worksheet. Make sure you complete the separate Assignments on time. Quizzes and tests may refer to work done in this Lab Worksheet; save your answers.

Before you get started - REMEMBER TO READ *ALL* THE WORDS

You must have an account on the Course Linux Server to do this lab. Log in to the server and use the shell. Review the Class Notes related to this worksheet as you work through it. Leave your work on the Linux server.

Refer back to Lab Three, *Using Standard Linux Commands I,* and ensure you understand everything written in the two sections near the beginning named **Linux Absolute and Relative Pathnames** and **Linux shell command syntax**. Review the definition of **basename** from the previous lab. Set your bash **`PS1`** shell prompt just as you did in the previous Lab.

Commands introduced and used in this Lab

- **`cat`** – display the contents of files without pagination (usually onto your screen)
- **`clear`** – to clear the screen of a terminal and put the cursor back at the top of the screen
- **`cp`** – copy one file to another, or copy one or more files into a directory
- **`find`** – to find pathnames (e.g. files or directories) by name, or by userid, or other criteria
- **`less`** (also **`more`**) – to page through a text file one screenfull at a time (better than **`cat`**)
- **`man`** – to get help for commands or system files or topics
- **`mv`** – move/rename pathnames, or move multiple pathnames into an existing directory
- **`rm`** – delete (remove) files (and entire directories of files **recursively**, with the **`-r`** option)
- **`sleep`** – do nothing (sleep) for some amount of time (pause a script)
- **`touch`** – to create an empty file and/or to update the file's date/time modified stamp

Log in to the Course Linux Server to do all commands in this lab. Set your bash **`PS1`** shell prompt to show your login name, computer name, and the **basename** of your current directory, just as you did in the previous Lab. Leave your finished work on the server; do not delete it when you are finished the worksheet. Keep your work.

Online help: Man pages revisited

The **`man`** command, short for Manual Pages, displays the manual page for the specified command. Man pages, as they are commonly referred to, contain all of the pertinent information on the basic command concepts, how to use the command, the command structure, basic options available for the command and how to use them, advanced options (if any), and related topics, in that order. The **`man`** command syntax is:

- **`man`** ***`command`*** - where ***`command`*** is the name of the command or thing you wish to learn about.

Examples: **`man ls ; man man ; man passwd ; man group ; man hosts`**

A text screen will show up with the information you requested - if it exists. You can then scroll up and down the man page using the **`up`** (↑) and **`down`** (↓) arrow keys and/or the **`[PgUp]`** and **`[PgDn]`** keys on your keyboard. You can also use the **spacebar** to scroll down one **screen**. Once you are done with the man page, simply type **`q`** for quit and you will exit the man page display. You can type **`q`** any time you want to exit the manual pages and you can type **`h`** or **`?`** for a **help** screen listing all the **other neat things** you can do while looking at this manual page. The most common thing to type is a **blank** (space), which displays the **next page** of the help file.

When you don't know the specific command for an operation, you can search the man page **titles** based on a keyword. (You can only search the title lines.) For this you need to specify the **`-k`** (keyword) option:

- **man -k *keyword*** (*multiple keywords will find more title lines for each word)*

Example: **man -k games** (*lists all man page title lines that contain the word **games**)*

1 Command: touch

The **touch** command updates the "last modified" time/date stamps on one or more existing files. It can also be used to create one or more **new**, **empty files**. See the manual page for more features.

Creating empty files and updating the modification time

Perform the following commands shown in **bold** type. Most commands will produce no output if they succeed.

Set your shell prompt: Before doing this lab, set your bash shell prompt to show your login name, the computer name, and the **basename** of your current working directory, as you did in the previous Lab.

```
[user@host  ]$ cd
[user@host ~]$ rm -rf lab4.1                    (remove this directory and everything inside it)
        (The above command will make a "clean slate" if you choose to restart this section from the start.)
[user@host ~]$ mkdir lab4.1                     (create a new, empty sub-directory)
[user@host ~]$ cd lab4.1                        (make this the new current working directory)
[user@host lab4.1]$ touch clock                 (create a new, empty file)
[user@host lab4.1]$ ls -li clock      (The option -l is lower-case letter L, not the digit 1)
```

a) Record **only** the **index** number and **time/date** stamp: _____

```
[user@host lab4.1]$ sleep 60                    (Waits for 60 seconds.  Read a book.)
[user@host lab4.1]$ touch clock                 (update the time stamp on the existing file)
[user@host lab4.1]$ ls -li clock                (the -l option is a letter, not a digit)
```

b) Record **only** the **index** number and new **time/date** stamp: _____

2 Command: cp (copy)

The **cp** (Copy) command makes a copy of files or directories. The syntax for the **cp** command is:

- **cp [*options*] *sources...* *destination***

where ***sources...*** is one or more files or directories and ***destination*** is either a file or a directory. If the destination is a directory, the file(s) will be copied into that directory using their **same names**. If you want to copy **directories**, you **must** use options such as **-r** or **-a**; otherwise, **cp** copies only source **files**.

```
[user@host  ]$ cd
[user@host ~]$ rm -rf lab4.2                    (remove this directory and everything inside it)
[user@host ~]$ mkdir lab4.2                     (create a new, empty sub-directory)
[user@host ~]$ cd lab4.2                        (make this the new current working directory)
[user@host lab4.2]$ touch a b c                 (create three new, empty files)
[user@host lab4.2]$ ls -i
```

a) Give the output of the last command, above: _____

```
[user@host lab4.2]$ mkdir mydir
[user@host lab4.2]$ ls -F                       (that is an UPPER CASE option letter)
```

b) Give the output of the last command, above: _____

```
[user@host lab4.2]$ cp a b c mydir
[user@host lab4.2]$ ls -i mydir
```

c) Give the output of the last command, above: _____

```
[user@host lab4.2]$ mkdir snack
[user@host lab4.2]$ touch snack/pie
[user@host lab4.2]$ cd snack
[user@host snack]$ touch apple
[user@host snack]$ cd ..
[user@host lab4.2]$ cp  snack/pie  snack/apple  mydir
[user@host lab4.2]$ ls mydir
```

d) Give the output of the last command, above: _____

```
[user@host lab4.2]$ mkdir A B C                    (these are UPPER CASE directory names)
```

e) What **command line** could you use to verify that the **three** directories have been **created?**

_____

```
[user@host lab4.2]$ touch  A/foo  B/bar  C/empty          (create three files)
[user@host lab4.2]$ cp A B C mydir          (try to copy A and B and C into mydir)
```

f) Record **one** of the messages displayed on the screen: _____

```
[user@host lab4.2]$ ls mydir                    (confirm that no directories were copied)
```

g) Why were the three source **A,B,C** directories **not copied** into destination directory **mydir**?

_____

```
[user@host lab4.2]$ cp -r A B C mydir          (the copy succeeds using this option!)
[user@host lab4.2]$ ls -R mydir                (that is an UPPER CASE option letter)
```

h) What **command line** could you use to see the **index number** and **date** of the **new copy** of file **empty**?

_____

```
[user@host lab4.2]$ mkdir -p parent/child              (remember what -p does?)
[user@host lab4.2]$ cp -r --parents parent/child mydir
```

i) What does the **--parents** option to **cp** do? (See "**man cp**".)

_____

j) Give the **absolute path** of the **new copy** of directory **child** after the above copy command creates it:

_____

3 Command: mv (move or rename)

The **mv** (Move or Rename) command moves (renames) files or directories. The renaming is very fast because the file data is *not* copied during a move/rename; only the *names* change (unless the move has to move the data to a different disk partition). Renaming is not a costly operation. The syntax for the **mv** command is:

- **mv [*options*] *sources...* *destination***

where ***sources...*** is one or more files or directories and ***destination*** is either a file or a directory. If the destination is a **directory**, the source files or directories will be moved (renamed) into that directory using their **same names**. If the destination is a **file**, only **one** source file is allowed to be moved (renamed). Examples:

- **mv file1 newfilename1**
- **mv directory1 newdirectoryname1**
- **mv file1 directory/newfilename1**
- **mv file1 file2 file3 directory**

```
[user@host  ]$ cd
[user@host ~]$ rm -rf lab4.3              (remove this directory and everything inside it)
[user@host ~]$ mkdir lab4.3
[user@host ~]$ cd lab4.3
[user@host lab4.3]$ touch A                          (create a single new, empty file)
[user@host lab4.3]$ ls -i                            (note the index number of A)
[user@host lab4.3]$ cp A foo                                    (copy the file)
[user@host lab4.3]$ mv A bar                                    (move the file)
[user@host lab4.3]$ ls -i                            (note the index numbers)
```

a) Looking at the index numbers: did moving file **A** to **bar** copy any data? _____

```
[user@host lab4.3]$ touch green blue orange          (create three new empty files)
[user@host lab4.3]$ mv green blue orange             (try to move/rename the files)
```

b) Record the error message: _____

```
[user@host lab4.3]$ mkdir colours
[user@host lab4.3]$ mv green blue orange colours
```

c) Give the **absolute** path of the file **blue**: _____

```
[user@host lab4.3]$ mkdir fans players arena                (three new directories)
[user@host lab4.3]$ touch  fans/me  players/you   (two new files inside two directories)
[user@host lab4.3]$ mv fans players arena          (move two directories into a third)
```

d) Give the new **absolute path** of the file named **you after** the above **mv** command has moved it:

_____

4 Command: `rm` (remove or delete files)

The **rm** (Remove or Delete) command removes (deletes) **files**. If the **-r** option is specified, it recursively deletes **directories** and **all** their contents. Unlike DOS, Windows, or OSX, a file or directory that is deleted with the **rm** command is **gone** (is **not** saved in a Recycle Bin) and not easily recovered. The syntax for the **rm** command is:

- **rm [*options*] *pathnames...***

Another useful option to **rm** is **-f** (force) that turns off any interactive prompts and most error messages. (We have been using "**rm -rf**" to completely recursively remove lab directories at the start of each section.)

Note: Most Unix/Linux shells let you type multiple commands on one line by separating them using the semi-colon character '**;**'. Type the **four** commands below, separated by **three** semi-colon characters:

```
[user@host  ]$ cd ; rm -rf lab4.4 ; mkdir lab4.4 ; cd lab4.4
[user@host lab4.4]$ mkdir sandbox sandbox/toybox          (create two directories)
[user@host lab4.4]$ touch toy1 toy2 toy3                  (create three empty files)
[user@host lab4.4]$ mv toy1 toy2 toy3 sandbox/toybox      (move all 3 files)
[user@host lab4.4]$ ls sandbox/toybox                     (you should see three toy files)
[user@host lab4.4]$ rmdir sandbox                         (try to remove the non-empty directory)
```

a) Record the error message: _____

```
[user@host lab4.4]$ rmdir -p sandbox        (try again to remove the non-empty directory)
```

b) Record the error message: _____

```
[user@host lab4.4]$ cp -a sandbox savebox    (save a full copy of sandbox in savebox)
[user@host lab4.4]$ ls -R savebox   (confirm that all sandbox has been copied to savebox)
[user@host lab4.4]$ rm -r sandbox          (recursively delete sandbox and everything in it)
[user@host lab4.4]$ ls                                    (confirm that sandbox is gone)
[user@host lab4.4]$ mv savebox sandbox                    (what does this do?)
```

c) Give the **absolute path** of the file **`toy2` after** the above commands are finished:

_____

```
[user@host lab4.4]$ cp -a sandbox/toybox sandbox    (recursive copy FAILS – why?)
```

d) Explain why the above copy fails: _____

_____

The **`-i`** option to **`rm`** will turn on "**Interactive**" mode, where you are prompted about *every* file being deleted:

```
[user@host lab4.4]$ cp -a sandbox/toybox .     (note the DOT ending this command line)
[user@host lab4.4]$ ls toybox                    (you should see three toy files in here)
[user@host lab4.4]$ rm -ri toybox              (answer yes to all the interactive questions)
```

5 Command: `cat` (catenate or show contents)

`cat` (Catenate, or Show) opens one or more files and catenates (shows) their contents. You can use it on any size file, but files are not be paginated and large files will scroll off your screen. Unlike **`less`**, **`cat`** won't warn you if you're about to **mess up** your terminal by displaying a **binary** format file. The syntax for **`cat`** is:

- **`cat [options] [file_list...]`**

Try these examples using **`cat`** and **`less`**:

```
[user@host ~]$ cat /etc/issue.net                 (this contains the network login banner)
[user@host ~]$ cat /etc/resolv.conf       (this contains the system DNS server IP addresses)
[user@host ~]$ cat /etc/issue.net /etc/resolv.conf /etc/issue.net
[user@host ~]$ cat /etc/services                    (many lines scroll off screen!)
[user@host ~]$ cat /etc/services | less          (don't use cat into less this way)
[user@host ~]$ less /etc/services                (the right way is to use less directly)
```

a) What option to **`cat`** shows non-printing characters? _____
b) What option to **`cat`** will number the output lines? _____
c) What option to **`cat`** will suppress repeated empty output lines? _____
d) "**`cat`**" backwards is "**`tac`**". What does the **`tac`** command do? _____
   Compare "**`cat /etc/resolv.conf`**" with "**`tac /etc/resolv.conf`**".

6 Command: `find` (find pathnames)

The **`find`** command recursively walks the directory tree structure, starting at a pathname given by the user, and finds (and usually prints) pathnames, based on *many* optional criteria. See the man page for the *many* options and features. The most common uses are (a) to find *all* pathnames under a directory, (b) find pathnames containing a particular **basename** pattern inside some starting directory, (c) find files *owned* by a particular userid, or (d) find files *modified* within some number of days:

- **`find [starting_directories...] -print`**
- **`find [starting_directories...] -name 'basename' -print`**
- **`find [starting_directories...] -user 'userid' -print`**
- **`find [starting_directories...] -mtime -days -print`**

Note that the name pattern is the ***`basename`***, found in any directory, starting from each of the the ***`starting_dirctories.`*** The ***`basename`*** patterns can include shell-GLOB-style path metacharacters such as "**`*`**" and "**`?`**". Note the unusual use of **full-words** used following *single*-dashes as **options** in this command! (Almost all other commands use *double* dashes for word-style options.) Examples:

- **`find . -print`** *(prints **all** the pathnames under the current directory)*
- **`find /etc -name 'passwd' -print`** *(print pathnames ending with basename **`passwd`**)*
- **`find /etc -name '*.conf' -print`** *(all pathnames ending in **`.conf`**)*
- **`find /bin -name '?ash' -print`** *(four-character basenames ending in '**`ash`**')*
- **`find /lib /usr/lib -name 'lib*.a' -print`** *(multiple starting directories)*
- **`find . -user root -print`** *(print only pathnames owned by this user)*
- **`find /var -mtime -30 -print`** *(print pathnames modified within last 30 days)*

a) What does the **`find`** option "**`-ls`**" do? (See "**`man find`**".)

_____

b) What does the **`find`** option "**`-type f`**" do? (See "**`man find`**".)

_____

c) What does the **`find`** option "**`-size 100M`**" do? (See "**`man find`**".)

_____

d) What does the **`find`** option "**`-size +100M`**" do (note the plus sign)? (*Hint*: Search the man page for the string "**`numeric arguments`**" which explains how numbers can be specified to **`find.`**)

_____

e) What command line recursively **finds** and displays only pathnames owned by userid **`idallen`** under the system directory **`/var/games`** ? (You should see at least two files.)

_____

f) What command line recursively **finds** and displays only pathnames ending in "**`log`**" in the system directory **`/etc`** (you will see many **Permission denied** messages in the output, as well as pathnames)?

_____

g) What command line recursively **finds** and displays only pathnames for things **bigger** than **500 Kilobytes** in the system directory **`/etc`** (you will see many **Permission denied** messages in the output)?

_____

7 Review exercise: `cd, mkdir, touch, mv, rm, cp, mkdir, find`

Keep a **reference list** of the commands used in every lab, along with the **options** used and what they **mean**. **Future** labs and tests will **expect** you to **remember** these command names and options.

Enter exactly the commands that are shown in **bold** below and note which commands produce **errors**. (There will be **three** errors; this is intentional.) Answer the questions following based **only** on these **review** commands. The **tilde** characters below have the same meaning as in the previous lab. (Go look!) Be precise in your typing!

1. `cd ; rm -rf ~/lab4.8`
2. `mkdir ~/lab4.8`
3. `cd ~/lab4.8`
4. `mkdir ./orchard`
5. `touch apple orange`
6. `mv orange orchard/lemon`
7. `rm orange`
8. `touch lettuce tomato cucumber`
9. `cp tomato lettuce garden`
10. `mkdir jardin forest`
11. `mv lettuce cucumber jardin`
12. `rmdir garden`
13. `touch lab4`
14. `cd orchard`
15. `cd ../../lab4.8/forest`
16. `mv ../lab4 ../tomato`

h) Give the command **number** that generated the error followed by the **full** and **exact** error message:

_____

_____

_____

i) What is the **absolute** path of the shell's current working directory after the **last** command (16), above?

_____

j) Give the **absolute pathname** of the one regular file **`lemon`** that is now in the directory named **`orchard`**:

_____

k) Give the relative path to the same **`lemon`** file from the **`forest`** directory:

_____

l) Give the relative path to the same **`lemon`** file from your own **HOME** directory:

_____

m) Give the relative path to the same **`lemon`** file from the directory called **`/home`**:

_____

n) Give the relative path to the same **`lemon`** file from the Linux ROOT directory:

_____

o) Give the relative path to the same **`lemon`** file from the directory called **`/root`**:

_____

p) List the **basenames** of **directories** that were successfully created (at any time) during the **review** exercise:

_____

_____

q) List the **absolute pathnames** of all **directories** that were **successfully** deleted during the **review** exercise:

_____

r) List the **absolute pathnames** of the five **regular files** still remaining anywhere under the directory **`lab4.8.`** Do **not** include the names of any **directories** or sub-directories – list only the five absolute **regular** file names located **anywhere** under the review directory **`lab4.8`**:

_____

_____

_____

_____

_____

s) What command line recursively **finds** and displays **all** pathnames under **`~/lab4.8`** ?

_____

t) Did you **READ *ALL* THE WORDS** in this lab? (Yes/No) _____